UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/020,656	10/29/2001	Tatsushi Inagaki	JP920000285	2851

7590 08/27/2004

Casey August
Intellectual Property Law Dept.
IBM Corporation
P.O. Box 218
Yorktown Heights, NY 10598

EXAMINER	
VO, TED T	

ART UNIT	PAPER NUMBER
2122	

DATE MAILED: 08/27/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/020,656	INAGAKI ET AL.	
	Examiner	Art Unit	
	Ted T. Vo	2122	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *29 October 2001*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-19* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-19* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☒ All b)☐ Some * c)☐ None of:

1.☒ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the communications filed on 10/29/2001.

Claims 1-19 are pending in the application.

Specification

2. The abstract of the disclosure is objected to because it does not comply with the guidelines in MPEP.

The abstract of the disclosure exceeds more than 150 words in length. The abstract should be in narrative form and generally limited to a single paragraph on a separate sheet within the range of 50 to 150 words (See MPEP § 608.01(b)). Correction is required.

Claim Objections

3. Claim 2 is objected to. A required format for a claim is that it starts with a capital word and ends with a period. After ending at "said program", with a period, Claim 2 continues with further limitations. This format is improper. Correction is required.

Claim Rejections - 35 USC § 102

4. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

5. Claims 1-19 are rejected under 35 U.S.C. 102(b) as being anticipated by Chambers et al., "Dependence Analysis for Java", 1999.

Given the broadest reasonable interpretation of followed claims in light of the specification.

As per Claim 1: Chambers discloses, "*A program optimization method for converting program source code written in a programming language into machine language comprising the steps of:*

analyzing a target program and detecting exception generative instructions, which may generate an exception, and exception generation detection instructions, which branches a process to an exception process when an exception occurrence condition is detected and an exception has occurred (See page 13, program in Fig.5 or Fig.6 provided with "PEI");

dividing said exception generation detection instructions into first instructions, for the detection of exception occurrence conditions (See page 8, section 3.5, second paragraph, "abstract locations for all variables [referring ALU instructions, section 3.3, page 8: *first instructions*] and memory locations [referring memory instructions, section 3.4, page 8]"), *and second instructions, for branching processes to exception processes when said exception occurrence conditions are detected* (See page 9, section 3.6 "for calls that raise exceptions" [also referring memory instructions, section 3.4, page 8: *second instructions*]); *and*

establishing dependencies among program instructions, so that when one of said exception occurrence conditions is detected the process is shifted from a first instruction to a second instruction, and so that when none of said exception occurrence conditions are detected the process is shifted from a first instruction to an exception generative instruction " (See the Graph generated in Fig.7, page 14; see page 5, third hyphen, "A PEI contains a pseudo assignment": *an exception generative instruction*).

As per Claim 2: Chambers discloses, "*The program optimization method according to claim 1, following said step for establishing said dependencies among the instructions, further comprising a step of: collecting multiple second instructions* (Call Instructions, Memory Instructions) *obtained based on multiple exception generation detection instructions detected within a predetermined range of said program* (Page

8: Section 3.5 defines rules for exception instructions: Exception type that includes ALU instructions and Memory instructions. Page 9: Call instructions: (*collecting multiple second instructions*)).

Thus, overall determination is performed based on the detection of said exception occurrence conditions by multiple first instructions, which are obtained based on multiple exception generation detection instructions." (This limitation concludes the functionality given from the above limitation of Claim 2; accordingly, the definitions of section 3.3, ALU Instructions, and section 3.4, Memory Instructions disclose the *overall determination.*)

As per Claim 3: Chambers discloses, "*The program optimization method according to claim 2, wherein said step of dividing said exception generation detection instructions includes a step of: generating flag setting instructions, as said first instructions, when said exception occurrence conditions are detected; and wherein said step of collecting said second instructions includes a step of: detecting the occurrence of exceptions, when flags are set for at least one of said multiple first instructions obtained based on said multiple exception generation detection instructions, and generating instructions for shifting processes so that said first instructions exchange positions with said second instructions*". For example, see page 16, second full paragraph, "load instructions (...) are now marked as PEIs": *Flags*).

As per Claim 4: Chambers discloses, "*The program optimization method according to claim 1, following said step of detecting said exception generative instruction, further comprising a step of: generating compensation code, when an exception generative instruction is accompanied by side effects, for maintaining order restrictions that are present before said exception generation detection instruction is divided and that concern said side effects*". For example, page 9, section 3.6, shows "side-effect analysis" that is capable of doing the functionality of the claim 4's limitation.

As per Claim 5: Chambers discloses, "*The program optimization method according to claim 1, following said step of setting said dependencies for the instruction, further comprising a step of: defining said first instruction as a conditional branch and allocating a predicate so as to reflect said conditional branch*". For example, section Introduction shows "Data dependence Analysis" that is capable of doing the functionality of the claim 5's limitation.

As per Claim 6: Chambers discloses, "*The program optimization method according to claim 1, following said step of setting said dependencies for the instructions, further comprising a step of: defining a first instruction as a conditional branch, and according to the result of code scheduling, generating a compensation code at a branching destination for said first instruction, so as to establish said order restrictions concerning said exception generation detection instruction before said exception generation detection instruction is divided into said first and said second instructions*". For example, see in section 2, pages 2-3; and section 3, Dependence Analysis, shows scheduling constraints of ALU instructions using abstract location that is capable of doing the functionality of the claim 6's limitation.

As per Claim 7: Regarding limitation: "*A compiler for converting the source code for a program written in a programming language into machine language and for optimizing said program comprising: a graph generator for analyzing a target program and for generating a graph showing the dependencies of operations in said program; a graph editing unit for editing said graph and for reducing order restrictions imposed on said operations due to the occurrence of an exception; and a code reproduction unit for generating program code that reflects said dependencies of said operations of said edited graph; wherein said graph editing unit detects an exception generative instruction, which may generate an exception, and an exception generation detection instruction, which branches a process to an exception process when an exception occurrence condition is detected and an exception has occurred, divides said detected exception generation detection instruction into a first instruction, which detects said exception occurrence condition, and a second instruction, which branches said process to said exception process when said exception occurrence condition is detected, and establishes a dependency among the instructions, so that the process is shifted from said first instruction in said graph to said second instruction when said exception occurrence condition is detected, or so that the process is shifted from said first instruction to said exception generative instruction when an exception occurrence condition is not detected*": the claim recites a compiler that has a graph generator, graph editing unit (referring to Fig. 7, page 14), and code reproducing unit (referring to Figs. 8, 9, pages 15-16). The claimed functionality implement the method recited in Claim. Claim 7 is rejected in the rationale set forth in the rejection of Claim 1.

As per Claim 8: Regarding: "*The compiler according to claim 7, wherein said graph editing unit removes from said graph order restrictions that are present before said exception generation detection instruction is divided and that concern said exception generative instruction, and order restrictions that are present before said exception generation detection instruction is divided and that precede said exception generation detection instruction.*" Chambers discloses such limitation. See page 16, first full paragraph, "The use of explicit exception test enables more reordering to be performed".

As per Claim 9: Claim 9 recites further limitation implemented to perform the steps as recited in Claim 2. Claim 9 is rejected in the rationale as set forth in the rejection of Claim 2.

As per Claim 10: Claim 10 recites further limitation implemented to perform the steps as recited in Claim 4. Claim 10 is rejected in the rationale as set forth in the rejection of Claim 4.

As per Claim 11: Claim 11 recites further limitation implemented to perform the steps as recited in Claims 5 and 6. Claim 11 is rejected in the rationale as set forth in the rejections of Claims 5 and 6.

As per Claim 12: Regarding: "*A compiler for converting the source code for a program written in a programming language into machine language and for optimizing the program comprising: an intermediate code generator for converting said source code into editable intermediate code; an optimization unit for optimizing said intermediate code; and a machine language code generator for using said obtained intermediate code to generate machine language code":* the limitation recites basic things of a compiler and the compiler disclosed by the reference is capable of doing as the same. Regarding: "*wherein said optimization unit analyzes said intermediate code for said program, transforms said program, within a predetermined range, so that an exception generative instruction that may generate an exception is executed before other instructions, establishes dependencies among the instructions of said program, so that, when an exception has been generated by said exception generative instruction, following the execution of said exception generative instruction the execution of instructions is inhibited, and converts an area in said program, within a predetermined range, so that the occurrence of an exception is detected when an exception is generated by at least one of multiple exception generative instructions, and a process is shifted to a corresponding exception process*": claim

recites optimization that establishes dependencies implemented as in the method recited in Claim 1. Claim 12 is rejected in the rationale set forth in the rejection of Claim 1.

As per Claim 13: Claim 13 recites further limitation implemented to perform the claimed functionality as recited in Claim 4. Claim 13 is rejected in the rationale as set forth in the rejection of Claim 4.

As per Claim 14: Claim 14 recites a computer implemented with a compiler that performs detecting exception as recited in Claim 1. Claim 14 is rejected in the same rationale as set forth in the rejection of Claim 1.

As per Claim 15: Claim 15 recites further limitation implemented to perform the claimed functionality as recited in Claim 2. Claim 15 is rejected in the rationale as set forth in the rejection of Claim 2.

As per Claim 16: Claim 16 recites a storage medium implemented with a compiler that performs the steps as recited in Claim 1. Claim 16 is rejected in the same rationale as set forth in the rejection of Claim 1.

As per Claim 17: Claim 17 recites further limitation implemented to perform the claimed functionality as recited in Claim 2. Claim 17 is rejected in the rationale as set forth in the rejection of Claim 2.

As per Claim 18: Claim 18 recites an apparatus implemented with a compiler that performs the steps as recited in Claim 1. Claim 18 is rejected in the same rationale as set forth in the rejection of Claim 1.

As per Claim 19: Claim 19 recites further limitation implemented to perform the claimed functionality as recited in Claim 2. Claim 19 is rejected in the rationale as set forth in the rejection of Claim 2.

Conclusion

6. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Choi et al., "Efficient and Precise Modeling of Exception for the Analysis of Java Programs", discloses Java exception model and implications for correct program analysis and optimization.

Dulong et al., "An Overview of the Intel™ IA-64 Compiler", discloses an Intel IA-64 architecture.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ted T. Vo whose telephone number is (703) 308-9049. The examiner can normally be reached on 8:00AM to 5:30PM.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (703) 305-4552. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

TED T. VO

TTV
Patent Examiner
Art Unit 2122
August 23, 2004